ATTACHMENT 6

LENS Access Technical Specification

Access by a Client Application

1. Overview

This document specifies the details of the interface that can be utilized by a Competitive Local Exchange Carrier (CLEC) to access the BellSouth Telecommunication's Local Exchange Negotiation System (LENS) from software emulating a Web Browser.

The LENS can be accessed directly by other computer systems bypassing the need for a Web Browser. This paper contains specifications for a methodology for using an application client in place of browser to communicate with and obtain information from the LENS Web server.

The LENS application will provide the following functionality related to the ordering of BellSouth Telecommunications services by CLECs.

- Street Address Validation
- Telephone Number Reservations
- Due Date Calculation
- Service Availability Inquires
- Creation of an Local Service Request (LSR)
- Customer Service Record Retrieval

The initial draft of this document will address only the first two functions listed above. Additional functions will be added to the document in later drafts until the document is complete. This document is based on the best information BellSouth Telecommunications (BST) has available at this time. It is, however, a DRAFT document and is subject to change in subsequent versions.

LENS is a Web-based application utilizing a Web Server to provide presentation of HTML code to remote browsers. It also includes a back-end application server that is accessed via CGI scripts. The remainder of this document provides technical details for the access to LENS from a client application other than a Web Browser.

2. General Interface Specifications

2.1 Interface Overview

1. This interface is based on emulation of a browser. The interface is designed to be utilized by either Netscape Navigator 3.x or Internet Explorer 3.0.

2. The client application will be required to utilize the HTTP protocol to obtain each of the pages in the user interface flow and to respond accordingly.

3. The interface requires the use of cookies. This mechanism is used to provide continuity between the pages.

4. All HTML pages are prefaced with a comment block. This block contains the data returned by the page in tag-value format. The general form of the comment block is as follows. The first five lines may be ignored.

```
HTTP/1.0 200 NeXT
Server: Netscape-Enterprise/2.0a
Date: current date
Content-type: text/html
Content-length: length of page
<!--
response=value
action=url
var1=value1
...
varn=valuen
-->
```

`action` delimits the URL of the CGI that must be called next. The tags `var1 ... varn` indicate the data values returned by the call. The remaining HTML page can be ignored by the application.

2.2 Connectivity

1. The network connectivity to the LENS application is the Internet suite of protocols (TCP/IP, etc.)

2. The protocol for the transfer of requests and responses is HTTP.

3. The connectivity to LENS can be through lan-to-lan connections, dial-up connections using PPP or through connections from the Internet.

2.3 Security

The security required will be dependent on the connectivity method utilized. Security for each of the three types of connectivity are discussed below

2.3.1 Lan-to-Lan

Security for the lan-to-lan connection assumes a trusted network on the other end and does not require additional network security. A CLEC wanting to use the Web Server over a lan-to-lan connection can obtain an application ID which can be utilized for all connections from the client application to the Web Server. The client application will have to include in the initial logon response information that can be used to identify the originating CLEC employee, if necessary, for auditing and trouble shooting. All access requires a registered IP Address.

2.3.2 Dial-up

BellSouth requires that any user making a dial-up connection be authenticated utilizing a Secure ID card. This card, in connection with a user ID, has to be utilized before the connection is made to the TCP/IP in-dial connections. In addition, the client application will be required to logon to the Web Server before beginning a session.

2.3.3 Internet

Any connection over the Internet will require the use of a security certificate obtained from a BellSouth designated certificate authority. In addition, all data sent back and forth will be encrypted using Secure Sockets Layer.

3. Application Specifications

This sections of the document details the HTML page flows for each application and defines the format, tags, etc. for each of the pages associated with the application. In addition any error messages will be documented.

After a to be determined period of time with no activity, the session is terminated with the following response.

```
<!--
response=Missing Session Error
-->
```

Note: The browser gets this message if the users session times out and they try to re-connect, but not otherwise. Otherwise, the application just drops the connection and it's up to the remote software to determine what happened.

3.1 Accessing the Pre-order Functionality

In order to reach the pre-order functionality, a session must be established. This requires a series of interactions with the application server. These interactions perform authentication and establish the pre-ordain session.

3.1.1 Page Flow

3.1.1.1 Initial Access

Open a connection to the specified server on the specified port. Send the following lines, each terminated by a carriage return and line feed, to the web server across the connection.

```
GET initial_application_url HTTP/1.0
User-Agent: agent name
```

3.1.1.2 Authentication

Open a connection to the specified server on the specified port. Send the following lines, each terminated by a carriage return and line feed, to the web server across the connection.

```
POST authentication_url HTTP/1.0
User-Agent: agent name
Content-type: application/x-www-form-urlencoded
Content-length: length(userid)+length(password)+9

1.0=userid&1.1=password
```

Note: *initial_application_url, agent name, userid* and *password* will be established after 3/31. `1.0` and `1.1` are the *names* associated with the HTML input fields. Read the output from standard in. Once the server has sent its response, it closes the connection.

3.1.1.3 Main Screen Access

Open a connection to the specified server on the specified port. Send the following lines, each terminated by a carriage return and line feed, to the web server across the connection.

```
POST mainscreen_url HTTP/1.0
User-Agent: agent name
Content-type: application/x-www-form-urlencoded
Content-length: 16
```

`1.2=Inquiry+Only`

Note: *`mainscreen_url`* is obtained from the `action` tag in the response from the initial access and authentication. Read the output from standard in. Once the server has sent its response, it closes the connection.

3.1.2 Data Format

The following tag-value pairs are returned by a successful access to each of the steps described in 3.1.1.

3.1.2.1 Initial Access Response

```
<!--
response=Authentication Screen
action=authentication_url
-->
```

3.1.2.2 Authentication Response

```
<!--
response=Main Screen
action=mainscreen_url
-->
```

3.1.2.3 Main Screen Response

```
<!--
response=Inquiry Screen
action=inquiryscreen_url
-->
```

3.1.3 Error Messages

3.1.3.1 Initial Access and Authentication Error

```
<!--
response=Authentication Error
action=initial_screen_url
-->
```

Note: The *initial_screen_url* is dynamically generated and is not identical to the *initial_application_url*

3.2 Street Address Validation

3.2.1 Page Flow

3.2.1.1 Street Address Validation Access

Open a connection to the specified server on the specified port. Send the following lines, each terminated by a carriage return and line feed, to the web server across the connection.

```
POST inquiryscreen_url HTTP/1.0
User-Agent: agent name
Content-type: application/x-www-form-urlencoded
Content-length: 27

1.0=Validate+Address&1.1=OK
```

Note: inquiryscreen_url is obtained from the `action` tag in the response from the main screen access. Read the output from standard in. Once the server has sent its response, it closes the connection.

3.2.1.2 Validate Address

Open a connection to the specified server on a specified port. Send the following lines, each terminated by a carriage return and line feed, to the web server across the connection.

```
POST validateAddr_url HTTP/1.0
User-Agent: agent name
Content-Type: application/x-www-form-urlencoded
Content-length: length_of_stdin_string

1.0=number&1.1=suffix&1.2=dir_prefix&1.3=thor_fare&1.4=dir_s
uffix&1.5=street_name&1.7=unit&1.9=elevation&1.11=structure&
1.12=city&1.13=state&1.14=phone_number&1.15=descriptive_addr
ess&1.16=route&1.17=box&1.18=occupant_name&1.20=Validate
```

Note: `validateAddr_url` refers to the action returned in the previous response. `agent name` will be established after 3/31. `1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.7, 1.9, 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18,` and `1.20`_are the *names* associated with the HTML input fields. The fields following these names are descriptive variable values to be substituted in the request. Include only those fields that you have data for in the stdin string. Because this value is variable, the `length_of_stdin_string` value will need to be computed for each request Once the server has sent its response, it closes the connection. The names of these fields are subject to change.

The values of the HTML field names described above have the following constraints:

`1.0`: Maximum length of 8 characters

`1.1`: Maximum length of 4 characters

`1.2`: One of the following values: N,E,S,W,NE,NW,SE,SW

`1.3`: One of the following values:
ALY,,ANX,ARC,AV,BEND,BLK,BDWK,BLVD,BR,BTM,BYP,CSWY,CTR,CIR,CV,
CT,CRES,CRK,CRSG,DR,ESPLND,EST,EXPWY,EXT,FRK,FRWY,GRDN,HBR,HL,
HLS,HT,HTS,HWY,HOLW,ISL,JCTN,LK,LDG,LN,LOOP,MNR,MKT,MT,MTN,NK,P
ASS,PATH,PK,PKWY,PR,PKE,PL,PLZ,PT,PD,PROM,PVT
DR,RDG,RD,RDWY,RT,ROW,RUN,SQ,STA,ST,TER,THRWY,TRC,TR,TRNPK,VLG
,WK,WAY,WHF,YD

`1.4`: One of the following values: N,E,S,W,NE,NW,SE,SW

`1.5`: Maximum length of 44 characters

`1.7`: Non-inclusive list of possible values: APT,LOT,RM,SLIP,SUIT,UNIT

`1.9`: Non-inclusive list of possible values: FLR

`1.11`: Non-inclusive list of possible values: BLDG,PIER,WNG

`1.12`: Maximum length of 32 characters

`1.13`: Standard two-character state abbreviations (capitalized)

`1.14`: Maximum length of 24 characters

`1.15`: Maximum length of 50 characters

`1.16`: Maximum length of 2 characters

`1.17`: Maximum length of 8 characters

`1.18`: Maximum length of 50 characters

`1.20`: field must be included with value of `Validate`

for viewing LSR errors *PlaceFirmOrderURL*, for creating an LSR, or *ChangeRequestURL* for changing an LSR.

3.1.3.2 Output Specification

Inquiry:

The returned inquiry screen contains the URL of the CGI that determines the preorder functionality to be executed and the choices of the functionality. The URL is in the FORM element and the data field choices and names are within the scope of this FORM.

```
<FORM method=post action="InquiryScreenURL">

<SELECT
name="inquiryOption"><OPTION></OPTION><OPTION>Validate
Address</OPTION><OPTION>Select Features and
Services</OPTION><OPTION>Reserve Telephone
Number(s)</OPTION><OPTION>View Installation
Calendar</OPTION><OPTION></OPTION></SELECT>

<INPUT size=24 maxlength=24 type=text
name="TelephoneNumber">

<INPUT size=2 maxlength=2 type=text name="StateSelection1">

<INPUT type=hidden name="MessageId" value="">

<INPUT type=hidden name="MessageText" value="">

<INPUT type=submit value="OK" name="OK">

<INPUT type=submit value="Cancel" name="Cancel">

</FORM>
```

View Firm Order Commit/Completion Notification :

See Section 3.6.

View Order Status:

See Section 3.7.

View LSR errors: .

See Section 3.8.

Place a Firm Order:

See Section 3.9.

Change an Existing Order:

See Section 3.10.

3.2 Input/Output Requirements for Street Address Validation

In this section the input/output requirements for street address validation are provided. Street address validation is the precursor to all other preorder functionality. The non-error flow is as follows address validation selection (3.2.1), address validation (3.2.2), address validation acknowledgment (3.2.3). In the case of any possible error response, the response and appropriate action are described.

3.2.1 Street Address Validation Request

Notify the cgi server that the application wishes to perform street address validation.

3.2.1.1 Input Specification

There are two possible input specification. The first simply notifies the cgi server that the application wishes to perform street address validation. The second passes a telephone number and state to the cgi server as the "address" to validate.

Send street address validation request:

```
POST InquiryScreenURL HTTP/1.0
User-Agent: agent name
Content-type: application/x-www-form-urlencoded
Content-length: 64

inquiryOption=Validate+Address&TelephoneNumber=&StateAbbr=&O
K=OK
```

Note: *InquiryScreenURL* is obtained from the `action` attribute in the form in the response from the main screen access.

Send telephone number to be validated.

```
POST InquiryScreenURL HTTP/1.0
User-Agent: agent name
Content-type: application/x-www-form-urlencoded
Content-length: 76

InquiryOption=Validate+Address&TelephoneNumber=TelephoneNumb
er&StateAbbr=stateAbbr&OK=OK
```

Note: *inquiryScreenURL* is obtained from the `action` attribute in the form in the response from the main screen access. *telephoneNumber* is a 10 digit telephone number containing no delimiting characters, and *stateAbbr* is a two letter USPS state abbreviation (all capitals). If no telephone number is sent, the telephone number and state

field names (`telephoneNumber, stateAbbr`) are included but no value is passed with them.

Return to Main Menu:

```
POST InquiryScreenURL HTTP/1.0
User-Agent: agent name
Content-type: application/x-www-form-urlencoded
Content-length: 13

Cancel=Cancel
```

Note: This action can be performed substituting the URL from the given form, for any form that contains a `Cancel` button.

3.2.1.2 Output Specification

The two inputs result in two different outputs. The first input returns an action that requires a street address or telephone number to be input. The second input causes the telephone number to be validated. It returns an action that expects an acknowledgment of the validated telephone number.

Action requiring an address to be input:

The URL for the next CGI and the required data fields are obtained from the following form in the page.

```
<FORM method=post action="ValidateAddrURL">

<INPUT size=8 maxlength=8 type=text
name="BasicStreetAddressNumber">

<INPUT size=44 maxlength=44 type=text
name="BasicStreetAddressStreetName">

<INPUT size=4 maxlength=4 type=text
name="BasicStreetAddressSuffix">

<SELECT name="BasicStreetDirPrefix">
<OPTION></OPTION><OPTION>E</OPTION><OPTION>N</OPTION><OPTION
>NE</OPTION><OPTION>NW</OPTION><OPTION>S</OPTION><OPTION>SE<
/OPTION><OPTION>SW</OPTION><OPTION>W</OPTION>

</SELECT>

<SELECT name="BasicStreetAddressThor">
<OPTION></OPTION><OPTION>ALY</OPTION><OPTION>ANX</OPTION><OP
TION>ARC</OPTION><OPTION>AV</OPTION><OPTION>BEND</OPTION><OP
TION>BLK</OPTION><OPTION>BDWK</OPTION><OPTION>BLVD</OPTION><
OPTION>BR</OPTION><OPTION>BTM</OPTION><OPTION>BYP</OPTION><O
PTION>CSWY</OPTION><OPTION>CTR</OPTION><OPTION>CIR</OPTION><
OPTION>CV</OPTION><OPTION>CT</OPTION><OPTION>CRES</OPTION><O
PTION>CRK</OPTION><OPTION>CRSG</OPTION><OPTION>DR</OPTION><O
```

```
PTION>ESPLND</OPTION><OPTION>EST</OPTION><OPTION>EXPWY</OPTI
ON><OPTION>EXT</OPTION><OPTION>FRK</OPTION><OPTION>FRWY</OPT
ION><OPTION>GRDN</OPTION><OPTION>HBR</OPTION><OPTION>HL</OPT
ION><OPTION>HLS</OPTION><OPTION>HT</OPTION><OPTION>HTS</OPTI
ON><OPTION>HWY</OPTION><OPTION>HOLW</OPTION><OPTION>ISL</OPT
ION><OPTION>JCTN</OPTION><OPTION>LK</OPTION><OPTION>LDG</OPT
ION><OPTION>LN</OPTION><OPTION>LOOP</OPTION><OPTION>MNR</OPT
ION><OPTION>MKT</OPTION><OPTION>MT</OPTION><OPTION>MTN</OPTI
ON><OPTION>NK</OPTION><OPTION>PASS</OPTION><OPTION>PATH</OPT
ION><OPTION>PK</OPTION><OPTION>PKWY</OPTION><OPTION>PR</OPTI
ON><OPTION>PKE</OPTION><OPTION>PL</OPTION><OPTION>PLZ</OPTIO
N><OPTION>PT</OPTION><OPTION>PD</OPTION><OPTION>PROM</OPTION
><OPTION>PVT</OPTION><OPTION>DR</OPTION><OPTION>RDG</OPTION>
<OPTION>RD</OPTION><OPTION>RDWY</OPTION><OPTION>RT</OPTION><
OPTION>ROW</OPTION><OPTION>RUN</OPTION><OPTION>SQ</OPTION><O
PTION>STA</OPTION><OPTION>ST</OPTION><OPTION>TER</OPTION><OP
TION>THRWY</OPTION><OPTION>TRC</OPTION><OPTION>TR</OPTION><O
PTION>TRNPK</OPTION><OPTION>VLG</OPTION><OPTION>WK</OPTION><
OPTION>WAY</OPTION><OPTION>WHF</OPTION><OPTION>YD</OPTION>

</SELECT>

<SELECT name="BasicStreetDirSuffix">
<OPTION></OPTION><OPTION>E</OPTION><OPTION>N</OPTION><OPTION
>NE</OPTION><OPTION>NW</OPTION><OPTION>S</OPTION><OPTION>SE<
/OPTION><OPTION>SW</OPTION><OPTION>W</OPTION>

</SELECT>

<SELECT name="SupplementalAddressUnitType">
<OPTION></OPTION><OPTION>APT</OPTION><OPTION>LOT</OPTION><OP
TION>RM</OPTION><OPTION>SLIP</OPTION><OPTION>SUIT</OPTION><O
PTION>UNIT</OPTION>

</SELECT>

<INPUT type=text size=10 name="SupplementalAddressUnitData">

<SELECT name="SupplementalAddressElevationType">
<OPTION></OPTION><OPTION>FL</OPTION>

</SELECT>

<INPUT type=text size=10
name="SupplementalAddressElevationData">

<SELECT name="SupplementalAddressStructureType">
<OPTION></OPTION><OPTION>BLDG</OPTION><OPTION>PIER</OPTION><
OPTION>WNG</OPTION>

</SELECT>

<INPUT type=text size=10
name="SupplementalAddressStructureData">

<INPUT backgroundcolor=ff1e2b maxlength=32 size=32 type=text
name="CommunityName">
```

```
<SELECT name="StateAbbr">
<OPTION</OPTION><OPTION>AL</OPTION><OPTION>FL</OPTION>GA<OPT
ION>KY</OPTION><OPTION>LA</OPTION><OPTION>MS</OPTION><OPTION
>NC</OPTION><OPTION>SC</OPTION><OPTION>TN</OPTION>

<INPUT size=50 maxlength=50 type=text
name="DescriptiveAddressName">

<INPUT size=2 maxlength=2 type=text
name="BasicStreetAddressRoute">

<INPUT size=8 type=text name="BasicStreetAddressBox">

<INPUT type=text size=24
name="TelephoneOrCircuitIdentifier">

<INPUT type=submit value="Validate" name="validate">

<INPUT type=submit value="Cancel" name="Cancel">

</FORM>
```

Action requiring an acknowledgment of a validated telephone number.
There are two forms returned on this page. The first is identical to the form returned in response to an address validation request. This form is populated with the value of the submitted telephone number and the submitted state. The second form is the acknowledgement of a validated address form. The second form contains the URL of the next CGI process and the name of the variables that must be passed to that CGI.

```
<FORM method=post action="SuccessfulValidationURL">

<INPUT type=hidden name="UnparsedStreetAddressStr"
value="StreetAddr">

<INPUT type=submit value="OK" name="OK">

<INPUT type=hidden value="Valid Address..."
name="hiddenValidAddr">

</FORM>
```

If this is the output received, proceed to section to 3.2.3 for the requirements of successful street address validation acknowledgment.

Return to Main Menu:

See 3.1.2.2.

3.2.1.3 Error Specification

This error can only arise if the input was a telephone number to be validated. The inquiry page is returned with an error message.

```
<FORM method=post action="InquiryScreenURL">
```

```
<SELECT name="1.0"><OPTION></OPTION><OPTION>Validate
Address</OPTION><OPTION>Select Features and
Services</OPTION><OPTION>Reserve Telephone
Number(s)</OPTION><OPTION>View Installation
Calendar</OPTION><OPTION></OPTION></SELECT>

<INPUT size=24 maxlength=24 type=text name="telephoneNumber"
value="sumbittedTN">

<INPUT size=2 maxlength=2 type=text name="stateAbbr"
value="submittedState">

<INPUT type=hidden name="MessageId" value="messageID">

<INPUT type=hidden name="MessageText" value="messageText">

<INPUT type=submit value="OK" name="OK">

<INPUT type=submit value="Cancel" name="Cancel">

</FORM>
```

The values *sumbittedTN* and *submittedState* are the telephone number and state that were submitted along with the validate address request.

3.2.2 Validate Address

Send the address to be validated to the cgi server.

3.2.2.1 Input Specification

Validate Street Address:

```
POST ValidateAddrURL HTTP/1.0

User-Agent: agent name

Content-Type: application/x-www-form-urlencoded

Content-length: length_of_stdin_string


BasicStreetAddressNumber=number&BasicStreetAddressStreetName
=street_name&BasicStreetAddressSuffix=suffix&BasicStreetDirP
refix=directional_prefix&BasicStreetAddressThor=thor_fare&Ba
sicStreetDirSuffix=dir_suffix&SupplementalAddressUnitType=un
it&SupplementalAddressUnitData=other_unit&SupplementalAddres
sElevationType=elevation&SupplementalAddressElevationData=ot
her_elevation&SupplementalAddressStructureType=structure&Sup
plementalAddressStructureData=other_structure&CommunityName=
city&StateAbbr=state&DescriptiveAddressName=descriptive_addr
ess&BasicAddressRoute=route&BasicStreetAddressBox=box&Teleph
oneOrCircuitIdentifier=telephone_number&validate=Validate
```

Note: *ValidateAddrURL* refers to the action returned in the previous response. *agent name* will be established after 3/31. The above string represents all possible inputs to the cgi. All field names must be included in the input. If the field is blank, no value is inserted. Because the input string changes with each address validation, its length must be calculated for each request.

The values of the form fields described contain the following values, and are subject to the following constraints:

`BasicStreetAddressNumber`: House number, maximum length of 8 characters.

`BasicStreetAddressStreetName`: Street name, a maximum length of 44 characters, any blanks must be replaced by a + .

`BasicStreetAddressSuffix`: A suffix.

`BasicStreetDirPrefix`: A directional prefix, one of the following values: `N,E,S,W,NE,NW,SE,SW.`

`BasicStreetAddressThor`: A thoroughfare abbreviation, one of the following values: `ALY, ANX, ARC, AV, BEND, BLK, BDWK, BLVD,BR, BTM, BYP, CSWY, CTR, CIR, CV, CT, CRES, CRK, CRSG, DR, ESPLND, EST, EXPWY, EXT, FRK, FRWY, GRDN, HBR, HL, HLS, HT, HTS, HWY, HOLW, ISL, JCTN, LK, LDG, LN, LOOP, MNR, MKT, MT, MTN, NK, PASS, PATH, PK, PKWY, PR, PKE, PL, PLZ, PT, PD, PROM, PVT, DR, RDG, RD, RDWY, RT,ROW, RUN, SQ, STA, ST, TER, THRWY, TRC, TR, TRNPK, VLG, WK, WAY, WHF, YD.`

`BasicStreetAddressDirSuffix`: A directional suffix, one of the following values: `N,E,S,W,NE,NW,SE,SW.`

`SupplementalAddressUnitType`: A living unit type, the non-inclusive list of possible values is `APT, LOT, RM, SLIP, SUIT, UNIT.`

`SupplementalAddressUnitData`: A living unit type not on the above list of values.

`SupplementalAddressElevationType`: A elevation indicator, the non-inclusive list of possible values is `FLR.`

`SupplementalAddressElevationData`: A elevation indicator, not on the above list.

`SupplementalAddressStructureType`: A structure type indicator, the non-inclusive list of possible values is `BLDG, PIER, WNG.`

`SupplementalAddressStructureData`: A structure type indicator not on the above list.

`CommunityName`: The city name, maximum length of 32 characters, all blanks in the city name must be replaced by the + sign.

`StateAbbr`: The USPS two-character state abbreviations (capitalized): `AL, FL, GA, KY, LA, MS, NC, SC, TN.`

`DescriptiveAddressName`: A descriptive address, maximum length of 50 characters, all blanks must be replaced by the + sign.

`BasicStreetAddressRoute`: A route number, maximum length of 2 characters.

`BasicStreetAddressBox`: A box number, maximum length of 8 characters.

`TelephoneOrCircuitIdentifier`: A telephone number, maximum length of 24 characters.

`validate`: A required field containing the value `Validate.`

3.2.2.2 Output Specification

Validate Street Address:

The page returned is identical to that returned when a telephone number is validated: two forms, one containing the valid address, and one containing the form that acknowledges the correct address. Again, it is the second form that contains the URL of the next CGI to be accessed and the names of the variables it expects as input. See Section 3.2.1.2 for the specification.

3.2.2.3 Error Specification

Invalid addresses fall into one of thirteen different error category. A different page is returned for each of these error categories. The pages differ only in the data that may be returned. The page always consists of two forms. The first is identical to that used to submit an address. This form is populated identically to the previously submitted address validation form. The URL in it is a *`RevalidateAddrURL`*. This is the URL to use in any revalidation. The second form contains a *`DoNothingAddrURL`* that is to be ignored, a set of hidden input fields containing data relevant to the error, and a `TEXTAREA` containing any possible valid addresses that can correctly replace the invalid address. The specification of the first form is given below. It is followed by all possible cases for the second form. To revalidate the address, follow the input specification in 3.2.2.1 substituting *`RevalidateAddrURL`* for *`ValidateAddrURL`*.

```
<FORM method=post action="RevalidateAddrURL">

<INPUT size=8 maxlength=8 type=text
name="BasicStreetAddressNumber" value="submittedNumber">

<INPUT size=44 maxlength=44 type=text
name="BasicStreetAddressStreetName" value="submittedStreet">

<INPUT size=4 maxlength=4 type=text
name="BasicStreetAddressSuffix" value="submittedSuffix">

<SELECT name="BasicStreetDirPrefix">
<OPTION>submittedDirPrefix</OPTION><OPTION>E</OPTION><OPTION
>N</OPTION><OPTION>NE</OPTION><OPTION>NW</OPTION><OPTION>S</
```

```
OPTION><OPTION>SE</OPTION><OPTION>SW</OPTION><OPTION>W</OPTI
ON>

</SELECT>

<SELECT name="BasicStreetAddressThor">
<OPTION>submittedThor</OPTION><OPTION>ALY</OPTION><OPTION>AN
X</OPTION><OPTION>ARC</OPTION><OPTION>AV</OPTION><OPTION>BEN
D</OPTION><OPTION>BLK</OPTION><OPTION>BDWK</OPTION><OPTION>B
LVD</OPTION><OPTION>BR</OPTION><OPTION>BTM</OPTION><OPTION>B
YP</OPTION><OPTION>CSWY</OPTION><OPTION>CTR</OPTION><OPTION>
CIR</OPTION><OPTION>CV</OPTION><OPTION>CT</OPTION><OPTION>CR
ES</OPTION><OPTION>CRK</OPTION><OPTION>CRSG</OPTION><OPTION>
DR</OPTION><OPTION>ESPLND</OPTION><OPTION>EST</OPTION><OPTIO
N>EXPWY</OPTION><OPTION>EXT</OPTION><OPTION>FRK</OPTION><OPT
ION>FRWY</OPTION><OPTION>GRDN</OPTION><OPTION>HBR</OPTION><O
PTION>HL</OPTION><OPTION>HLS</OPTION><OPTION>HT</OPTION><OPT
ION>HTS</OPTION><OPTION>HWY</OPTION><OPTION>HOLW</OPTION><OP
TION>ISL</OPTION><OPTION>JCTN</OPTION><OPTION>LK</OPTION><OP
TION>LDG</OPTION><OPTION>LN</OPTION><OPTION>LOOP</OPTION><OP
TION>MNR</OPTION><OPTION>MKT</OPTION><OPTION>MT</OPTION><OPT
ION>MTN</OPTION><OPTION>NK</OPTION><OPTION>PASS</OPTION><OPT
ION>PATH</OPTION><OPTION>PK</OPTION><OPTION>PKWY</OPTION><OP
TION>PR</OPTION><OPTION>PKE</OPTION><OPTION>PL</OPTION><OPTI
ON>PLZ</OPTION><OPTION>PT</OPTION><OPTION>PD</OPTION><OPTION
>PROM</OPTION><OPTION>PVT</OPTION><OPTION>DR</OPTION><OPTION
>RDG</OPTION><OPTION>RD</OPTION><OPTION>RDWY</OPTION><OPTION
>RT</OPTION><OPTION>ROW</OPTION><OPTION>RUN</OPTION><OPTION>
SQ</OPTION><OPTION>STA</OPTION><OPTION>ST</OPTION><OPTION>TE
R</OPTION><OPTION>THRWY</OPTION><OPTION>TRC</OPTION><OPTION>
TR</OPTION><OPTION>TRNPK</OPTION><OPTION>VLG</OPTION><OPTION
>WK</OPTION><OPTION>WAY</OPTION><OPTION>WHF</OPTION><OPTION>
YD</OPTION>

</SELECT>

<SELECT name="BasicStreetDirSuffix">
<OPTION>submittedDirSuffix</OPTION><OPTION>E</OPTION><OPTION
>N</OPTION><OPTION>NE</OPTION><OPTION>NW</OPTION><OPTION>S</
OPTION><OPTION>SE</OPTION><OPTION>SW</OPTION><OPTION>W</OPTI
ON>

</SELECT>

<SELECT name="SupplementalAddressUnitType">
<OPTION>submittedUnitOption</OPTION><OPTION>APT</OPTION><OPT
ION>LOT</OPTION><OPTION>RM</OPTION><OPTION>SLIP</OPTION><OPT
ION>SUIT</OPTION><OPTION>UNIT</OPTION>

</SELECT>

<INPUT type=text size=10 name="SupplementalAddressUnitData"
value="submittedUnitStr">
```

```
<SELECT name="SupplementalAddressElevationType">
<OPTION>submittedElevationOption</OPTION><OPTION>FL</OPTION>

</SELECT>

<INPUT type=text size=10
name="SupplementalAddressElevationData"
value"submittedElevationStr">

<SELECT name="SupplementalAddressStructureType">
<OPTION>submittedStructureOption</OPTION><OPTION>BLDG</OPTIO
N><OPTION>PIER</OPTION><OPTION>WNG</OPTION>

</SELECT>

<INPUT type=text size=10
name="SupplementalAddressStructureData"
value="submittedStructureStr">

<INPUT backgroundcolor=ff1e2b maxlength=32 size=32 type=text
name="CommunityName" value="submittedCommunity">

<SELECT name="StateAbbr">
<OPTION>submittedState</OPTION><OPTION>AL</OPTION><OPTION>FL
</OPTION>GA<OPTION>KY</OPTION><OPTION>LA</OPTION><OPTION>MS<
/OPTION><OPTION>NC</OPTION><OPTION>SC</OPTION><OPTION>TN</OP
TION>

<INPUT size=50 maxlength=50 type=text
name="DescriptiveAddressName" value="submittedDescr">

<INPUT size=2 maxlength=2 type=text
name="BasicStreetAddressRoute" value="submittedRoute">

<INPUT size=8 type=text name="BasicStreetAddressBox"
value="submittedBox">

<INPUT type=text size=24 name="TelephoneOrCircuitIdentifier"
value="submittedTN">

<INPUT type=submit value="Validate" name="validate">

<INPUT type=submit value="Cancel" name="Cancel">

</FORM>
```

Each string prefixed by *submitted* represents the value that was previously submitted.

The thirteen forms corresponding to the different errors are shown below.

Error Type AHN

This error arises when there is no assigned house number.

```
<FORM method=post action="DoNothingAddrURL">

<INPUT type=hidden name="MessageId" value="messageID">

<INPUT type=hidden name="MessageText" value="messageText">
```

```
<INPUT type=hidden name="BasicStAddNumberStr"
value="StreetNumber">

<INPUT type=hidden name="BasicStAddSuffixStr"
value="Suffix">

<INPUT type=hidden name="BasicStAddRouteStr" value="Route">

<INPUT type=hidden name="BasicStAddBoxStr" value="Box">

<INPUT type=hidden name="CommunityAbbreviationStr"
value="CommunityAbbreviation">

</FORM>
```

Error Type C

This error arises when no match at all has been found and therefore the user must resend all information with corrections for a new validation attempt.

```
<FORM method=post action="DoNothingAddrURL">

<INPUT type=hidden name="MessageId" value="messageID">

<INPUT type=hidden name="MessageText" value="messageText">

<INPUT type=text name="CommunityCrossBoundaryStateField"
value="CommunityCrossBoundaryState">

</FORM>
```

Error Type E

This error arises when the community cannot be matched.

```
<FORM method=post action="DoNothingAddrURL">

<INPUT type=hidden name="MessageId" value="messageID">

<INPUT type=hidden name="MessageText" value="messageText">

<INPUT type=hidden name="UnparsedStreetAddressStr"
value="StreetAddr">

<TEXTAREA rows=5 cols=50 name="CurrentCommunityText">

choice1

...

choicen

</TEXTAREA>

</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for the community name. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type F

This error arises when there is no exact match found on the street name.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="CommunityNameStr"
value="communityName">
<TEXTAREA rows=5 cols=50 name="CurrentCommunityStreetText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for the street. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type G

This error arises when ?.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="CommunityNameStr"
value="communityName">
<TEXTAREA rows=5 cols=50
name="CurrentDescriptiveAddressText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any descriptive addresses that are partial matches for the submitted address. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type H

This error arises when ?.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="AddressStringStr"
value="AddressString">
<INPUT type=hidden name="CommunityNameStringStr"
value="CommunityString">
<INPUT type=hidden name="StateAbbrStr" value="StateString">
<TEXTAREA rows=5 cols=50 name="CurrentStreetRangeText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for the street number. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type I

This error arises when ?.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="AddressStringStr"
value="AddressString">
<INPUT type=hidden name="CommunityNameStringStr"
value="CommunityString">
<INPUT type=hidden name="DescriptiveAddrStringStr"
value="DescriptiveAddrString">
```

```
<INPUT type=hidden name="StateAbbrStr" value="StateString">
<TEXTAREA rows=5 cols=50 name="UnitStringText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for the unit. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type J

This error arises when ?.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="AddressStringStr"
value="AddressString">
<INPUT type=hidden name="CommunityNameStringStr"
value="CommunityString">
<INPUT type=hidden name="DescriptiveAddrStringStr"
value="DescriptiveAddrString">
<INPUT type=hidden name="StateAbbrStr" value="StateString">
<TEXTAREA rows=5 cols=50
name="SupplementalAddressStringText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for the supplemental address. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type K

This error arises when ?.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="CommunityNameStr"
value="communityName">
<TEXTAREA rows=5 cols=50 name="AddressStringText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for the ?. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type M

This error arises when ?.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="ParsedAddressStr"
value="ParsedAddress">
<INPUT type=hidden name="CommunityNameStr"
value="CommunityName">
<INPUT type=hidden name="StateAbbrStr"
value="StateAbbreviation">
</FORM>
```

Error Type N

This error arises when ?.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
```

```
<INPUT type=hidden name="TelephoneNumberStr"
value="TelephoneNumber">
<TEXTAREA rows=5 cols=50 name="AddressStringText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for ?. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

Error Type O

This error arises when there is no exact match found on the street name.

```
<FORM method=post action="DoNothingAddrURL">
<INPUT type=hidden name="MessageId" value="messageID">
<INPUT type=hidden name="MessageText" value="messageText">
<INPUT type=hidden name="CommunityNameStr"
value="CommunityName">
<TEXTAREA rows=5 cols=50 name="AddressStringText">
choice1
...
choicen
</TEXTAREA>
</FORM>
```

Note: *choice1, ..., choicen* are the strings representing any partial matches for ?. Each choice appears on a separate line, and is delimited by a newline in the text (not an HTML tag).

3.2.3 Successful Street Address Validation Acknowledgment

Agree to the successful validation.

3.2.3.1 Input Specification

```
POST SuccessfulValidationURL HTTP/1.0
User-Agent: agent name
Content-type: application/x-www-form-urlencoded
Content-length: 5

OK=OK
```

Note: *SuccessfulValidationURL* is obtained from the `action` tag in the response from the validate address

3.2.3.2 Output Specification

See 3.1.3.2.

3.3 Telephone Number Reservation

In this section the input/output requirements for telephone number reservation are provided. Street address validation must have occurred before telephone number reservation can occur. This must be included in the telephone number reservation process. It is accomplished either by sending the cgi server a telephone number and state along with the request to reserve telephone numbers or by requesting telephone number reservation without including a telephone number. In this case, the user will be sent to the street address validation sequence described in 3.2. In the case in which a telephone number and state are sent with a telephone number reservation request, the non-error flow is as follows: telephone number reservation selection (3.3.1), telephone number type selection (3.3.2), telephone number selection (3.3.3), telephone number reservation(3.3.4). In the case in which the telephone number reservation request is not accompanied by a telephone number the flow includes street address validation as follows: telephone number reservation selection (3.3.1), street address validation (see Section 3.2), telephone number type selection (3.3.2), telephone number selection (3.3.3), telephone number reservation (3.3.4). Telephone number selection (3.3.3) is repeated until the desired set of numbers have been selected. This section is structured following the first sequential flow since the address validation flow is documented above.

3.3.1 Telephone Number Reservation Selection

Send the CGI server the request to select and reserve telephone numbers. This request may be accompanied by a telephone number and state.